Queue and Interface

CS200 Recitation 3

Announcements

- WA 1 is graded, come get it during recitation. Grades also on RamCT
- Answers to last week's induction problems posted to RamCT discussion board.

Tip of the Week

- The website Wolfram|Alpha [1] can be a great help with math. `http://www.wolframalpha.com/`. Forgot to mention it last week, sorry.
- In the terminal, the **`top`** command shows you a list of processes ordered by processor use. It's a good way to see what is using a machine.

```
top - 15:47:41 up 2 days, 23:30,  2 users,  load average: 4.02, 3.76, 3.71
Tasks: 173 total,   2 running, 171 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.2%us,  0.1%sy, 38.7%ni, 60.7%id,  0.0%wa,  0.2%hi,  0.0%si,  0.0%st
Mem:  16436944k total, 16158908k used,   278036k free,   108044k buffers
Swap: 16787888k total,    89984k used, 16697904k free,  3632172k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3668 user1     39  19 11.4g  10g 2144 S 381.9 69.7  12440:28 big-program
16876 user2     20   0  118m 4848  860 S  2.0  0.0   0:01.20 sshd
17383 user2     20   0  688m  62m  33m S  2.0  0.4   0:00.96 chrome
    1 root      20   0 36248 4924 1808 S  0.0  0.0   0:02.18 systemd
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.10 kthreadd
```

  Notice: %CPU, %MEM, and NI columns. Default sorts list by highest %CPU. Positive values in the NI column means that process has a 'nice' value and is running at lowered priority. In this example there is an intensive process running with nice value 19, it is lower priority than everything else on the machine and is not interfering with performance. This is mandatory etiquette for intensive processes.

  - If you are trying to use a super-slow computer, and another machine is not available, let the lab operator know.
  - An intensive process is not necessarily the cause of a slow machine.

Interfaces

What is an interface?

- A contract between a class and the outside world
- The class must provide functionality specified in the interface

- The buttons on a TV remote control are a good example of an interface. They provide communication between the device and the world, and more importantly they specify exactly what the device can do.

A Java Interface:

- ONLY CONTAINS: constants, method signatures, and nested types.
- CANNOT BE INSTANTIATED: it must be implemented by a class or extended by another interface

How are they used? Application Programming Interfaces (APIs)

- Very common way to distribute software
- Everything is compiled, the interface(s) are published.
- Users use the interface to access everything
- Under-the-hood changes (bug fixes, optimization, new stuff) are invisible to users.
- Users do not have to know or care what is happening behind the interface

`How to make an interface?` Much like a class but it is called an interface instead:

```
public interface cs200_queue {
        public boolean isEmpty();          /* return true if queue is empty */
        public int size();                 /* return number of items in the queue */
        public void enqueue(Object item); /* add item to back of queue */
        public Object dequeue();           /* remove object from front of queue */
}
```

`How does a class implement an interface?` By specifying it when creating the class with the **`implements`** keyword:

```
public class best_class implements cs200_queue {
  //code goes here
}
```

Exercise

Today's exercise is to build two queue classes implementing the above cs200_queue interface. These two queues will both implement the functionality of a queue, but will do so with different underlying storage. **`Get recit3.tar off the class website`**, it has a copy of the interface and a main class for testing.

- **`LinkedList_queue`** will build a queue out of a java.util.LinkedList object. You should be able to use mostly methods of Linked List to implement the queue
- **`Array_queue`** will build a queue on top of an array. Since an array does not have methods, you will have to do things more manually. Where in the array should you put the front of the queue (used in dequeue())?. Where should you put the back? You will probably need to keep track of some array indices to locate the front and back of the queue. Further, when objects are removed from the queue you may need to move other objects around to keep the queue continuous.

Both queue objects will need a constructor in addition to the interface. For LinkedList_queue, this constructor probably will not do much. For Array_queue the constructor needs to take an integer argument (max size of the queue) and make a new array which can store that many objects.
When you are done, tar up your code and submit it using checkin, this is Recit 3:

- Monday: **`~cs200/bin/checkin R3L01 R3L01.tar`**
- Tuesday: **`~cs200/bin/checkin R3L02 R3L02.tar`**
- Wednesday: **`~cs200/bin/checkin R3L03 R3L03.tar`**
- Thursday: **`~cs200/bin/checkin R3L04 R3L04.tar`**

Finally, **`sign the attendance sheet.`**

References

[1] Wolfram Alpha LLC. Wolfram|alpha. `http://www.wolframalpha.com/`, January 2012.